Δίκτυα Υπολογιστών

Άσκηση 1

1 Key-Value Store

Ένα *key-value store* είναι μια υπηρεσία στην οποία μπορούμε να αποθηκεύουμε ζεύγη `<key,value>`, και στη συνέχεια να ανακτούμε την τιμή (value) βάσει του αντίστοιχου κλειδιού (key).

Στην άσκηση αυτή καλείστε θα υλοποιήσετε ένα απλό δικτυακό key-value store βασισμένο στο πρωτόκολλο TCP. Αυτό θα απαρτίζεται από έναν server που θα υλοποιεί το key-value store, και έναν client που θα το χρησιμοποιεί. Ο server πρέπει να δέχεται συνδέσεις των clients και να εκτελεί τις εντολές τους για αποθήκευση (*put*) και ανάκτηση (*get*) δεδομένων. Δεδομένου του ότι στην άσκηση αυτή θα εστιάσουμε μόνο στο δικτυακό τμήμα, ο server θα αποθηκεύει τα δεδομένα μόνο στη μνήμη του, δηλαδή δεν χρειάζεται να τα αποθηκεύει σε αρχεία ή σε βάση δεδομένων. Τέλος, τα keys και τα values έχουν αλφαριθμητικές τιμές (δηλ., είναι strings).

2 Πρωτόκολλο Εφαρμογής (Application Protocol)

Η επικοινωνία μεταξύ του server και του client ορίζεται αυστηρά βάσει του εξής πρωτοκόλλου. Μόλις ο client συνδεθεί με τον server, τού στέλνει μία ή περισσότερες εντολές. Μια εντολή ξεκινάει με ένα byte που ορίζει το είδος της λειτουργίας: η τιμή 103 σημαίνει *get* και η τιμή 112 σημαίνει *put*. Οποιοσδήποτε άλλος κωδικός εντολής θεωρείται λάθος και ο server θα πρέπει σε αυτή την περίπτωση να τερματίζει τη σύνδεση. Αμέσως μετά τον κωδικό εντολής ο client στέλνει το key (σε περίπτωση get) ή το key ακολουθούμενο από το value (σε περίπτωση put). Και το key και το value πρέπει να σταλούν ως null-terminated strings.

Αυτό ολοκληρώνει την αποστολή μιας εντολής, και ο client δεν πρέπει να στείλει τίποτα άλλο, εκτός και αν θέλει να στείλει και άλλη εντολή στην ίδια σύνδεση. Σε αυτή την περίπτωση, αμέσως μετά το τελευταίο '\0' της τρέχουσας εντολής ξεκινάει την επόμενη εντολή, στέλνοντας τον κωδικό της και τα απαραίτητα στοιχεία (key ή key και value). Ο client μπορεί να στείλει *όσες* εντολές θέλει σε μία σύνδεση, χωρίς κανέναν περιορισμό.

Κατά τη λήψη μιας εντολής, ο server δρα ως εξής. Σε περίπτωση εντολής get, ο server ψάχνει στην τοπική του μνήμη για το ζητούμενο key και απαντάει στον client. Το πρώτο byte αυτής της απάντησης δηλώνει αν το κλειδί βρέθηκε (κωδικός 102) ή δεν βρέθηκε (κωδικός 110). Αν βρέθηκε, το σχετικό value πρέπει να σταλεί ως null-terminated string αμέσως μετά τον αριθμό 102. Αν δεν βρέθηκε, ο κωδικός 110 ολοκληρώνει την απάντηση, δεν πρέπει να σταλεί κανένα άλλο byte στα πλαίσια αυτής την εντολής.

Και ο server και ο client πρέπει να κλείνουν τη σύνδεση σε περίπτωση που ο άλλος δεν τηρεί πιστά το πρωτόκολλο. Όμως, εντολές που προηγούνται του λάθους πρέπει να απαντώνται κανονικά πριν τη διακοπή της σύνδεσης (δηλαδή τα αποτελέσματα πρέπει να έχουν σταλεί στον client πριν τη διακοπή). Γενικά ο client μπορεί να στείλει όσες εντολές θέλει και να κλείσει τη σύνδεση όποτε θέλει, ενώ ο server δεν παίρνει ποτέ την πρωτοβουλία να κλείσει τη σύνδεση, εκτός από την περίπτωση λάθους του πρωτοκόλλου εκ μέρους του client. Προφανώς οποιαδήποτε αλληλουχία εντολών, συνδέσεων, σφαλμάτων, και διακοπών εκ μέρους του client δεν πρέπει να επηρεάζει τη σωστή λειτουργία του server.

3 Υλοποίηση του Client

Ο client πρέπει να υλοποιηθεί στο αρχείο `client.c`, και πρέπει να δέχεται έναν μεταβλητό αριθμό παραμέτρων (command-line arguments). Η πρώτη παράμετρος είναι το hostname του server, και το

δεύτερο είναι το port στο οποίο ακούει ο server. Οι υπόλοιπες παράμετροι πρέπει να περιλαμβάνουν μία ή περισσότερες εντολές. Για μια εντολή get, οι παράμετροι είναι `get <key>`, ενώ για μια εντολή put είναι `put <key> <value>`. Για τις εντολές get ο client πρέπει να τυπώνει στην οθόνη το value που επέστρεψε ο server ακολουθούμενο από new line, ή ένα σκέτο new line (δηλ. μία κενή γραμμή) αν το ζητούμενο key δεν βρέθηκε. Για εντολές put δεν πρέπει να γράφει τίποτα στην οθόνη.

Μια τυπική ακολουθία εκτελέσεων του client θα μπορούσε να είναι η ακόλουθη:

```
$ ./client diogenis.ceid.upatras.gr 5555 put city Patra
$ ./client diogenis.ceid.upatras.gr 5555 get city
Patra
$ ./client diogenis.ceid.upatras.gr 5555 put country Greece get country
Greece
$ ./client diogenis.ceid.upatras.gr 5555 get city put TK 26504 get country
Patra
Greece
$ ./client diogenis.ceid.upatras.gr 5555 get city get unknown_key get TK
Patra

26504
$
```

4 Υλοποίηση του Server

Ο server πρέπει να δέχεται συνδέσεις σε *όλα* τα interfaces του υπολογιστή στον οποίο τρέχει, σε ένα TCP port που παρέχεται από τον χρήστη σαν παράμετρος. Πρέπει να υλοποιήσετε τέσσερις (4) εκδοχές του server, που θα ονομάζονται `serv1`, `serv2`, `serv3`, και `serv4`. Κάθε εκδοχή βασίζεται σε διαφορετική αρχιτεκτονική λειτουργίας, ως εξής:

1. **Iterative server**: Ο `serv1` πρέπει να υλοποιηθεί ως iterative structure, δηλαδή να δέχεται έναν client κάθε φορά. Στο listen backlog πρέπει να δοθεί η τιμή 5 (που είναι και η default τιμή).

2. **On-demand forking server**: Ο `serv2` πρέπει να υποστηρίζει περισσότερους από έναν clients ταυτόχρονα, δημιουργώντας μια διεργασία-παιδί για κάθε νέο client που συνδέεται (one-process-per-client).

3. **Preforking server**: Ο `serv3` πρέπει να χρησιμοποιεί pre-forking, δηλαδή να φτιάχνει κατά την εκκίνησή του έναν αριθμό από διεργασίες (worker processes), οι οποίες να αναλαμβάνουν την εξυπηρέτηση των πελατών. Ο αριθμός των worker processes ορίζεται από τον χρήστη ως μια δεύτερη παράμετρος μετά το port.

4. **Multi-threaded server**: Ο `serv4` πρέπει να λειτουργεί ως μία και μόνο διεργασία που δημιουργεί ξεχωριστό thread ανά client που συνδέεται. Δηλαδή είναι παρόμοια η λειτουργία του με τον `serv2`, αλλά αντί για processes πρέπει να φτιάχνει threads για νέους clients.

Όλοι οι servers παίρνουν από μία παράμετρο, που δηλώνει το TCP port, με εξαίρεση τον `serv3` ο οποίος παίρνει μια δεύτερη παράμετρο, τον αριθμό των preforked worker processes. Π.χ., οι servers θα μπορούσαν να εκτελεσθούν ως εξής (ο καθένας φυσικά διατηρεί και διαχειρίζεται το δικό του key-value store, εντελώς ανεξάρτητα από τους άλλους):

```
$ ./serv1 5555
$ ./serv2 5556
$ ./serv3 5557 10
$ ./serv4 5558
$
```

Η υλοποίηση του key-value store μέσα σε κάθε server πρέπει να γίνεται βάσει των συναρτήσεων `put()` και `get()` όπως ορίζονται στο `keyvalue.h` (Σχήμα 1).

```
#ifndef __KEYVALUE_H
#define __KEYVALUE_H

/*
 * Returns the value associated with the key, or NULL
 * if the key is not in the store.
 */
char *get(char *key);

/*
 * Puts the <key,value> pair into the store.
 * If the key already existed, its previous value
 * is discarded and replaced by the new one.
 */
void put(char *key, char *value);

#endif
```

Σχήμα 1: `keyvalue.h`

Συγχρονισμός Οι servers σας, με εξαίρεση τον (serv1), πρέπει να επιτρέπουν σε πολλούς clients να είναι ταυτόχρονα συνδεδεμένοι και να στέλνουν εντολές. Για την ανάγνωση τιμών, δεν υπάρχει πρόβλημα η ίδια τιμή να σταλεί σε δύο ή περισσότερους clients ταυτόχρονα. Για παράλληλη εγγραφή τιμών όμως, ή για ανάγνωση μιας τιμής την ώρα που κάποιος άλλος client την εγγράφει, πρέπει να λάβετε τα μέτρα σας ώστε να μην δημιουργηθούν race conditions.

5 Οδηγίες παράδοσης

Στείλτε μας συνημμένα στη διεύθυνση **spyros+hw1@ceid.upatras.gr** το `client.c`, τους τέσσερις servers (αρχεία `serv[1234].c`), καθώς και ένα `report.pdf` στο οποίο εξηγείτε σύντομα τη λογική κάθε server.

Το subject του email ΠΡΕΠΕΙ να είναι αποκλειστικά και μόνο ο Αριθμός Μητρώου σας (π.χ. 1234), χωρίς έξτρα κενά, χωρίς "ΑΜ: 1234", χωρίς "Re:", χωρίς οτιδήποτε άλλο πέρα από τα τέσσερα αριθμητικά ψηφία του ΑΜ σας. Το email πρέπει να σταλεί από τον λογαριασμό σας στο CEID. Θα λάβετε άμεσα επιβεβαίωση για το email σας.

Μπορείτε να στείλετε την υλοποίησή σας όσες φορές θέλετε. Η **τελευταία** υλοποίηση που θα λάβουμε θα είναι και αυτή που θα βαθμολογηθεί. Όλες οι προηγούμενες θα αγνοηθούν. Αν συνεχίσετε να στέλνετε υλοποιήσεις μετά το τέλος της προθεσμίας, για κάθε μέρα καθυστέρησης θα χρεώνεστε έναν βαθμό στη συγκεκριμένη άσκηση. Ισχύουν πέντε (5) μέρες χάριτος.

Μην ξεχάσετε το **report.pdf**!

Καλή επιτυχία!